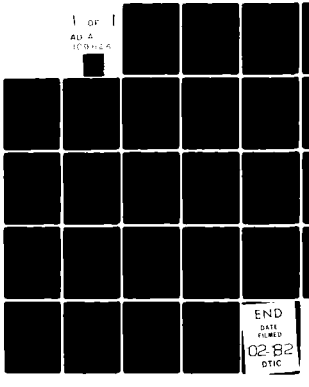
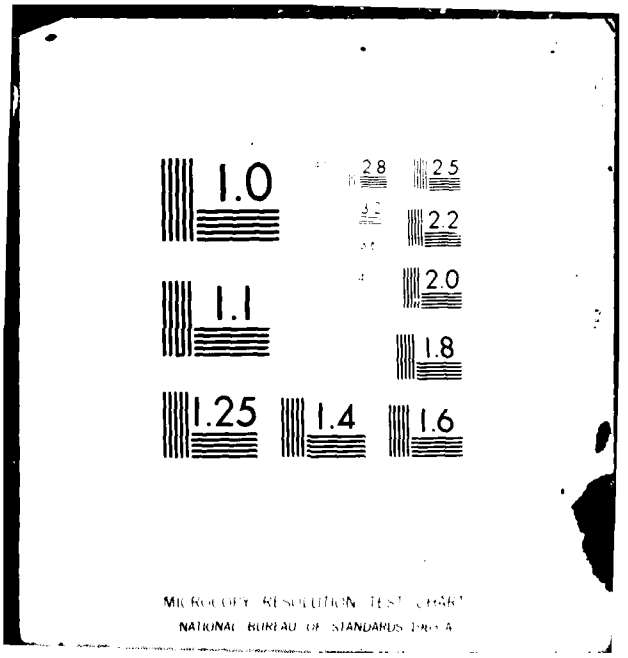
AD-A109 626 MASSACHUSETTS INST OF TECH CAMBRIDGE LAB FOR INFORMA--ETC F/6 12/1
NOTES ON OPTIMAL ROUTING AND FLOW CONTROL FOR COMMUNICATION NET--ETC(U)
DEC 81 D P BERTSEKAS N00014-75-C-1183
UNCLASSIFIED LIDS-R-1169 NL




1.0
2.8
2.5
2.2
2.0
1.1
1.8
1.25
1.4
1.6
MICROCOPY RESOLUTION TEST CHART
NATIONAL BUREAU OF STANDARDS 1963-A

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

REPORT DOCUMENTATION PAGE		READ INSTRUCTIONS BEFORE COMPLETING FORM
1. REPORT NUMBER	2. GOVT ACCESSION NO. AD-A109626	3. RECIPIENT'S CATALOG NUMBER
4. TITLE (and Subtitle) Notes on Optimal Routing and Flow Control for Communication Networks		5. TYPE OF REPORT & PERIOD COVERED Technical Report
		6. PERFORMING ORG. REPORT NUMBER LIDS-R-1169
7. AUTHOR(s) Dimitri P. Bertsekas (12)		8. CONTRACT OR GRANT NUMBER(s) ARPA Contract ONR-N00014-75-C-1183
9. PERFORMING ORGANIZATION NAME AND ADDRESS M.I.T. Laboratory for Information and Decision Systems Cambridge, MA 02139		10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS Program Code No. 5T10 ONR Identifying No. 049-383
11. CONTROLLING OFFICE NAME AND ADDRESS Defense Advanced Research Projects Agency 1400 Wilson Boulevard Arlington, VA 22209		12. REPORT DATE December, 1981
		13. NUMBER OF PAGES 56
14. MONITORING AGENCY NAME & ADDRESS (if different from Controlling Office) Office of Naval Research Information Systems Program Code 437 Arlington, VA 22217		15. SECURITY CLASS. (of this report) Unclassified
		15a. DECLASSIFICATION/DOWNGRADING SCHEDULE

16. DISTRIBUTION STATEMENT (of this Report)

Approved for public release; distribution unlimited.

17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)

18. SUPPLEMENTARY NOTES

19. KEY WORDS (Continue on reverse side if necessary and identify by block number)

20. ABSTRACT (Continue on reverse side if necessary and identify by block number)

DD FORM 1 JAN 73 1473 EDITION OF 1 NOV 65 IS OBSOLETE

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

December 1981 LIDS-R-1169

NOTES ON OPTIMAL ROUTING AND FLOW CONTROL FOR COMMUNICATION NETWORKS[†]

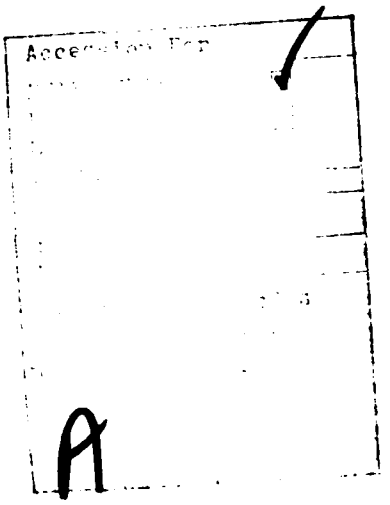
by

Dimitri P. Bertsekas*

[†]This research was conducted at the M.I.T. Laboratory for Information and Decision Systems with partial support provided by Defense Advanced Research Projects Agency under contract No. ONR-N00014-75-C-1183.

*The author is with the Massachusetts Institute of Technology, Laboratory for Information and Decision Systems, Room No. 35-210, Cambridge, Mass. 02139.

TABLE OF CONTENTS



1. Introduction

The main purpose of routing and flow control in a communication network is, roughly speaking, to keep delay per message withing an acceptable level while minimizing the amount of offered traffic that is rejected by the network due to its inability to handle it. These two objectives are clearly contradictory so a good routing and flow control scheme must strike a balance between the two. It should also take into account a number of other issues such as fairness for all users, the possibility that the network topology can be altered due to unexpected link or node failures, and the fact that the statistics of offered traffic change with time.

In these notes we consider some aspects of routing and flow control for long-haul wire data networks in which the communication resource is scarce (as opposed to local networks such as Ethernet where it is not), and where there are no issues of contention resolution due to random access of a broadcast medium (as in some satellite, local, and packet radio networks). We place primary emphasis on optimal procedures since these offer a more sound philosophical basis than heuristic schemes and also provide a yardstick for measuring the effectiveness of other methods.

We consider primarily the subject of routing in a quasistatic offered load environment. By routing we mean the set of decisions regarding the outgoing links to be used for routing data at each subnetwork node. By quasistatic environment (see [1]) we mean a situation where the offered traffic statistics for each origin-destination pair change slowly over time and furthermore individual offered traffic sample functions do not exhibit frequently large and persistent deviations from their averages. A typical quasistatic network is one accomodating a large number of

interactive users for each origin-destination pair and in which the law of large numbers approximately takes hold. In such an environment it is valid to base routing decisions on average levels of traffic input flows which can be estimated from past history measurements.

Situations where the quasistatic assumption is not valid are typically characterized by the presence of few and large users that can by themselves overload the network over brief periods of time if left uncontrolled. We then talk of a need to provide dynamic routing. By this we mean short term adjustment of routes to adapt to the instantaneous network state which includes instantaneous traffic input rates as well as queue lengths. Dynamic routing is a subject that is insufficiently well understood at present and should probably be studied in combination with flow control. We will not consider it further in these notes.

While there are situations where routing can be considered in isolation from flow control, in other cases the interactions between routing and flow control are so strong that they cannot be ignored in a meaningful analysis. By flow control we mean the set of decisions regarding the amount of traffic to be admitted in the network for each origin-destination pair or each user pair conversation. It is intuitively clear that if data is routed efficiently within the network then more traffic can be admitted into the network without violating the users' dissatisfaction threshold. Therefore incremental changes in routing can be expected to have an effect on the amount of traffic that flow control should allow to enter the network. On the other hand routing changes should take into account the concurrent effects of flow control if they are to be effective. The resulting coupling between routing and flow control can be quite complex

and only recently there has been substantial progress towards understanding it. Some of the most important work in this area [10],[11] is described in the last section where a combined routing and flow control optimization problem is formulated. It turns out that this problem is essentially the same as the optimal routing problem and can be solved by simple adaptation of the type of algorithms described in these notes. Another related subject of considerable current interest is routing and flow control of real-time data--that is data that, if not delivered within a specified time delay, becomes useless. Digitized voice is a prime example of such data. We refer to [12] for related work.

2. Routing Variables in Quasistatic Routing

Traffic congestion in a quasistatic data network can be reasonably well evaluated in terms of the arrival rates at each of the transmission queues. There is one such queue per directed link in the network and its arrival rate is referred to as the total flow of the link. For a link (i,k) we use the symbol F_{ik} to denote the corresponding total flow. This flow is measured in data units per sec where the data units can be bits, packets, messages, etc. Sometimes it is meaningful to measure flow in units that are assumed to be directly proportional to data units per sec such as virtual circuit calls traversing the link.

Congestion is typically measured in terms of some function of the total flows F_{ik}. For example

$$\sum_{(i,k)} D_{ik}(F_{ik}) \tag{1}$$

is a frequently used measure of congestion where $D_{ik}(F_{ik})$ represents (an approximation to) the average number of messages in queue or under transmission at link (i,k) when the flow is F_{ik}. A frequently used formula is

$$D_{ik}(F_{ik}) = \frac{F_{ik}}{C_{ik}-F_{ik}} \tag{2}$$

where C_{ik} is the transmission capacity of the (i,k) transmission line measured in the same units as F_{ik}. This is based on the hypothesis that each queue behaves as an M/M/1 queue and is referred to as the Kleinrock independence assumption. While this hypothesis is almost never true in practice the expression (1), (2) represents a useful measure of performance since it expresses qualitatively the fact that congestion sets in when a

flow F_{ik} approaches the corresponding link capacity C_{ik}. Another useful measure of congestion is given by

$$\max_{(i,k)} \{\frac{F_{ik}}{C_{ik}}\} \tag{3}$$

i.e. maximum link utilization. A computational study [3] has shown that it typically makes little difference whether the objective function (1)-(2) or (3) is used for optimizing routing. This is particularly true for heavily loaded networks where computational results show that optimal routing with respect to one objective function [(1)-(2) or (3)] is within very few (1-3) percentage points of being optimal with respect to the other.

It is useful to break down the total link flows into the portions that have common destination. Thus for a link (i,k) we denote by $f_{ik}(j)$ the flow in the transmission queue (i,k) of data units destined for node j. Clearly we have

$$F_{ik} = \sum_{j} f_{ik}(j). \tag{4}$$

Furthermore conservation of flow holds at each node in the form

$$\sum_{(m,i)} f_{mi}(j) + r_i(j) = \sum_{(i,k)} f_{ik}(j), \quad \forall\ i,j \text{ with } i \neq j \tag{5}$$

The right side of (5) represents the total outgoing flow from node i that is destined for j, while the left side represents the total incoming flow into i from other nodes that is destined for j, plus the terms $r_i(j)$ which represents flow entering the network at node i and destined for j. By adding (5) for all destinations j we see that there is also conservation

of total flows at each node i.

The objective of the routing algorithms that we will consider can be loosely stated as follows:

Given a set of external traffic intputs $\{r_i(j)\}$ [cf. (5)] find a "desirable" corresponding set of total flows $\{F_{ik}\}$.

Let us leave aside for the moment the question of how we measure "desirability" of the set of total flows $\{F_{ik}\}$ [i.e. which objective function such as (1) or (3) we use], and concentrate on the instruments (or controls) in our disposal for influencing the values of $\{F_{ik}\}$. These are called the routing variables. We first discuss link routing variables and then consider path routing variables.

The routing variable of link (i,k) with respect to destination j is defined by

$$\phi_{ik}(j) = \frac{f_{ik}(j)}{\sum_m f_{im}(j)} \tag{6}$$

for nodes i such that $\sum_m f_{im}(j) > 0$, and represents the fraction of flow arriving at i and destined for j which is routed through link (i,k). We have

$$\sum_k \phi_{ik}(j) = 1, \quad \phi_{ik}(j) \geq 0, \quad \forall\, k,i,j\,,\ i \neq j \tag{7}$$

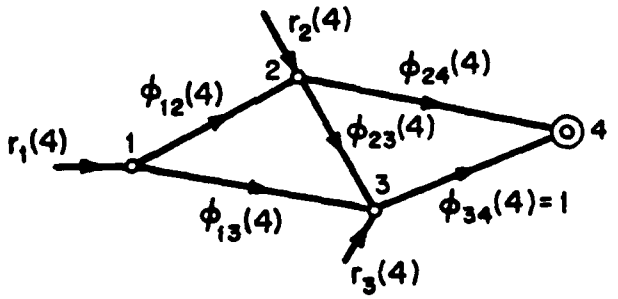
For nodes i and j such that $\sum_m f_{im}(j) = 0$ any set of numbers $\phi_{ik}(j)$ satisfying (7) can serve as corresponding link routing variables. Note that routing variables of the form $\phi_{jm}(j)$ (i.e. i=j) do not make sense and are not defined.

A set of link routing variables $\{\phi_{ik}(j)\}$, i.e. a set of numbers satisfy-

ing (7), is said to be acyclic and destination oriented (ADO for short) if the following condition holds:

There is no destination node j and directed cycle (i,k_1), $(k_1,k_2),\ldots,$ (k_m,i) not containing j along which the routing variables $\phi_{ik_1}(j)$, $\phi_{k_1k_2}(j)$, $\ldots,\phi_{k_mi}(j)$ are all positive.

A little thought shows that this condition implies that given any pair of nodes i and j there exists a directed path $\{(i,k_1), (k_1,k_2),\ldots,$ $(k_m,j)\}$ from i to j along which the routing variables $\phi_{ik_1}(j)$, $\phi_{k_1k_2}(j),\ldots,$ $\phi_{k_mj}(j)$ are positive. It should be clear that in data networks we are primarily interested in routing variable sets that are ADO for otherwise data would be allowed to travel on a loop with an obvious inefficiency resulting.† Another easily seen fact is that a set of external traffic inputs $\{r_i(j)\}$ and a set of ADO routing variables $\{\phi_{ik}(j)\}$ define uniquely a corresponding set of flows $\{f_{ik}(j)\}$ via equations (5) and (6). Furthermore if r represents the vector of traffic inputs and ϕ the vector of ADO routing variables, the corresponding vector f of flows can be defined in terms of some function $f(\phi,r)$ which depends only on the topology of the network. For example



†We assume here implicitly that the objective function is a nondecreasing function of each total link flow.

in the network shown in the figure (links (i,k) with $\phi_{ik} = 0$ are not shown) we have

$$f_{12}(4) = r_1(4)\phi_{12}(4)$$

$$f_{13}(4) = r_1(4)\phi_{13}(4)$$

$$f_{23}(4) = [r_1(4)\phi_{12}(4) + r_2(4)]\phi_{23}(4)$$

$$f_{24}(4) = [r_1(4)\phi_{12}(4) + r_2(4)]\phi_{24}(4)$$

$$f_{34}(4) = [r_1(4)\phi_{12}(4) + r_2(4)]\phi_{23}(4) + r_1(4)\phi_{13}(4) + r_3(4)$$

Clearly the form of the function $f(\phi,r)$ can be quite complicated and non-linear but this fact does not cause significant algorithmic difficulties. For example it is easy to construct an algorithm which for given ϕ and r generates the corresponding flow vector $f(\phi,r)$, and the corresponding set of total flows $\{F_{ik}(\phi,r)\}$. We can then pose the optimal routing problem of finding a set of ADO routing variables which for fixed and given set of inputs $\{r_i(j)\}$ minimizes an objective function of the total flows such as (1) or (3).

An alternate but equivalent formulation of the optimal routing problem is obtained by considering <u>path routing variables</u> in place of link routing variables. For each pair $w = (i,j)$ of distinct nodes i and j [also called an origin-destination (or OD) pair], denote by P_w the set of all simple directed paths from i to j. For each OD pair $w = (i,j)$ the input $r_i(j)$, also written r_w, is to be divided into individual <u>path flows</u> h_p, where $p \epsilon P_w$, satisfying

$$r_w = \sum_{p \epsilon P_w} h_p \ , \quad h_p \geq 0 \ , \qquad \forall \ p \epsilon P_w . \tag{8}$$

Given a set of path flows satisfying (8) the corresponding path routing variables for OD pairs w with $r_w > 0$ are defined by

$$\xi_p = \frac{h_p}{r_w} \quad , \qquad \forall\ p \epsilon P_w \tag{9}$$

and simply represent the fractions of input routed along the corresponding paths. It follows that path routing variables satisfy

$$\sum_{p \epsilon P_w} \xi_p = 1, \qquad \xi_p \geq 0, \qquad \forall\ p \epsilon P_w.$$

Clearly a set of path routing variables together with a set of inputs $\{r_w\}$ defines uniquely a corresponding set of path flows via (8). These path flows in turn define uniquely a corresponding set of link flows obtained by adding, for each link and destination the path flows that traverse the link and correspond to that destination.

A conclusion is that an optimal routing problem can be posed whereby, for a fixed and given set of OD pair inputs $\{r_w\}$, we wish to find a set of path routing variables which minimizes an objective function of total flows.

It is important to realize that <u>the two formulations of the routing problem in terms of path routing variables and ADO link routing variables are equivalent</u>. The reason is that given a set of inputs $\{r_w\}$ and a set of path routing variables $\{\xi_p\}$ there exists a set of ADO link routing variables $\{\phi_{ik}(j)\}$ with the property that $\{\xi_p\}$ and $\{\phi_{ik}(j)\}$ generate identical sets of link flows. The set $\{\phi_{ik}(j)\}$ is unique except for nodes i and destinations j for which the total flow $\sum_m f_{mi}(j)$ is zero.

The reverse is not entirely true. Given a set of ADO link routing

variables there is at least one but possibly more than one corresponding sets of path routing variables that generate identical sets of link flows. Proving these facts is a simple and instructive exercise for the reader.

3. Implementation by Means of Routing Variables

We think of a routing algorithm as a process whereby the set of routing variables is modified occassionally according to some rules. Before getting into the details of various types of routing algorithms it is worth considering briefly the practical implementation of a set of routing variables. The chief means for doing this are the routing tables kept at each node. At this point we must distinguish as to whether the network uses datagrams or virtual circuits.

In a network using datagrams each message or packet (including packets of the same pair of users) is routed independently of the others. For the purposes of routing the only information that the message carries is the destination ID number. Suppose we desire to implement a set of link routing variables $\{\phi_{ik}(j)\}$ in a datagram network. One way of doing this is for each node i to maintain a routing table whereby for each destination j and each outgoing link (i,k) the routing variable $\phi_{ik}(j)$ is stored together with the actual fraction $\hat{\phi}_{ik}(j)$ of the number of data units (messages, bits etc) for destination j actually routed along link (i,k) during the time elapsed since the latest routing variable update. When a new message arrives node i looks up its destination j, assigns the message to the outgoing link (i,k) for which the ratio $\phi_{ik}(j)/\hat{\phi}_{ik}(j)$ is largest, and updates the corresponding fractions $\hat{\phi}_{ik}(j)$. There are other possible implementations which may differ in minor details but the idea is clear. Traffic is metered to keep track of the actual fractions of the number of data units travelling along each outgoing link and the choice of route is designed to match as close as possible the actual fractions with the target fractions given by the link routing variables. Each time

the link routing variables change, each node incorporates the new values in the routing tables and reinitializes the actual fractions $\hat{\phi}_{ik}(j)$ to some positive values, for example $\hat{\phi}_{ik}(j) = \frac{1}{m}$ for all links (i,k) with $\phi_{ik}(j) > 0$ where m is the number of these links. Note that the link routing variables actually used for the construction of the routing tables could themselves be obtained by first determining (using the "master" routing algorithm) a set of path routing variables for each OD pair and then computing the (essentially unique) corresponding set of link routing variables.

In a network using virtual circuits all the messages belonging to the same conversation travel along the same path during the full duration of a conversation [By conversation here we mean a connection between two users (persons or machines) engaged in message exchange through the network.] The path is set up at the beginning of the conversation when one of the two users requests a connection with the other similarly as for ordinary telephone calls. Once a path is set up each node along the path keeps in a table sufficient information to ensure that messages of each conversation follow the same route. Routing variables come into play by affecting the choice of route at the beginning of the conversation. There are several ways that this can be done.

Suppose first that path routing variables $\{\xi_p\}$ are available for each OD pair. Each node i keeps a count of the number of virtual circuit calls that use each one of the paths with itself as the origin. It also maintains the fractions ξ_p of the number of calls on each path p divided by the total number of calls on paths that have the same origin and destination as path p. When a new call request is received at node i for some destination j, node i calculates the path p for the OD pair (i,j) for which $\xi_p/\hat{\xi}_p$ is

largest, assigns the call on that path, and updates the corresponding fractions $\hat{\xi}_p$. The actual path is established by sending along the path a setup packet with the sequence of links of the path stamped on it. The fractions $\hat{\xi}_p$ are of course adjusted each time a virtual circuit is terminated. As new calls are established and old calls are terminated the values of the actual fractions $\hat{\xi}_p$ drift gradually towards their desired values ξ_p specified by the routing variables even if these values happen to be substantially different at times due to changes in ξ_p. It is of course also possible to change forcibly at any time the routes of some virtual calls in order to make the actual and desired fractions $\hat{\xi}_p$ and ξ_p close to each other and this must be done each time a node or link fails thereby disrupting some of the physical communication paths.

Consider next the case of a virtual circuit network where we wish to implement a set of ADO link routing variables $\{\phi_{ik}(j)\}$. Each node i maintains the fractions $\hat{\phi}_{ik}(j)$ of the number of virtual circuits passing through node i, having j as destination and routed through link (i,k), divided by the total number of virtual circuits passing through i and destined for j. When a new call request is received at some origin node m with destination j, the node m sends a path finding packet along the link k for which the ratio $\phi_{mk}(j)/\hat{\phi}_{mk}(j)$ is largest. When the path finding packet reaches a new node, say i, it is subsequently routed along the link k for which $\phi_{ik}/\hat{\phi}_{ik}(j)$ is largest, until it reaches the destination j. At this point the path of the new virtual circuit call will have been established. Note that this method of using ADO link routing variables is very similar to the one described earlier for datagrams. Indeed we may view a datagram as a degenerate form of virtual circuit involving a single

packet transmission. If this point of view is adopted the link routing variable based method of implementation for virtual circuits just described reduces to the one described earlier for datagrams.

There are a number of variations on the implementation methods described above. For example $\{\hat{\xi}_p\}$ or $\{\hat{\phi}_{ik}(j)\}$ could represent fractions of flow rather than virtual circuits etc. The main point to keep in mind is that <u>while the choice of virtual circuits versus datagrams and the corresponding implementation of the routing strategy are important practical design issues, they are largely decoupled from the conceptual issues of how one should choose and update routing variables, i.e., how one should design the routing algorithm</u>.

4. Characterization of Optimal Routing Variables

Suppose we are given a directed network with set of nodes N and set of links L. Let W be a collection of ordered node pairs referred to as origin-destination (OD) pairs. For each OD pair $w \in W$ we are given a positive number r_w respresenting rate of input into the network from origin to destination measured in data units per sec. Let P_w be the set of all simple directed paths joining the OD pair w, and for each path $p \in P_w$ let us denote by h_p the flow on path p in data units per sec. We have thus the constraint

$$\sum_{p \in P_w} h_p = r_w \quad , \quad h_p \geq 0, \quad \forall\, p \in P_w,\ w \in W \tag{1}$$

For an OD pair $w \in W$, a path $p \in P_w$ and a link $(i,k) \in L$ we denote

$$\delta_p(i,k) = \begin{cases} 1 & \text{if path p contains link } (i,k) \\ 0 & \text{otherwise} \end{cases} \tag{2}$$

Then the total flow on each link $(i,k) \in L$ is given in terms of the individual path flows by means of the linear expression

$$F_{ik} = \sum_{w \in W} \sum_{p \in P_w} \delta_p(i,k) h_p. \tag{3}$$

In the remainder of these notes we concentrate on an objective function of the form

$$\sum_{(i,k) \in L} D_{ik}(F_{ik}) \tag{4}$$

and the problem of finding the set of path flows $\{h_p\}$ that minimize this

objective function subject to the constraints (1) and (3). Reference [3] considers the problem of minimizing the maximum link utilization $\max\{ \frac{F_{ik}}{C_{ik}} \mid (i,k) \in L\}$ by using algorithms that bear close relation to those used for minimizing the objective function (4).

By eliminating the total flows F_{ik} from the objective function (4) we can write the problem as

$$\text{minimize} \quad \sum_{(i,k)\in} D_{ik}\Big[\sum_{w\in W} \sum_{p\in P_w} \delta_p(i,k) h_p \Big]$$

$$\text{subject to} \quad \sum_{p\in P_w} h_p = r_w , \quad \forall\, w\in W \tag{5}$$

$$h_p \geq 0 , \quad \forall\, p\in P_w,\ w\in W.$$

We assume that each D_{ik} is a twice differentiable function of the scalar variable F_{ik} and is defined in an interval $[0,C_{ik})$ where C_{ik} is either a positive number (typically representing the capacity of the link) or else is $+\infty$. The first and second derivatives of D_{ik} are denoted D'_{ik} and D''_{ik} and are assumed strictly positive for all $F_{ik}\in[0,C_{ik})$. This implies in particular that D_{ik} is a convex monotonically increasing function of F_{ik}.

We wish to characterize optimal solutions of problem (5) and then derive algorithms for its solution. Note that an optimal set of path flows $\{h_p^*\}$ yields immediately a set of optimal path routing variables $\{\xi_p^*\}$ via the formula

$$\xi_p^* = \frac{h_p^*}{r_w} ,$$

so this formulation of the routing problem is geared towars yielding optimal path routing variables. On the other hand we have seen that each set of path routing variables yields an optimal set of link routing variables. An alternative is to formulate the routing problem directly in terms of link routing variables. We refer to the papers [1] and [2] for a presentation of possibilities along these lines.

A characterization of optimal solutions of the routing problem (5) is obtained by specializing the following general necessary and sufficient condition for optimality:

Lemma: Let $f: R^n \to R$ be a differentiable convex function on the n-dimensional Euclidean space R^n, and let X be a convex subset of R^n. Then $x^* \in X$ is an optimal solution of the problem

$$\begin{aligned} &\text{minimize} \quad f(x) \\ &\text{subject to } x \in X \end{aligned} \tag{6}$$

if and only if

$$\nabla f(x^*)^T (x - x^*) \geq 0, \quad \forall\, x \in X, \tag{7}$$

where $\nabla f(x^*)$ is the gradient vector of f at x^* and superscript T denotes transpose.

Proof. Assume x^* is an optimal solution of (6), and for every $x \in X$ consider the function $g(\alpha) = f[x^* + \alpha(x - x^*)]$ of the scalar variable α. Then $g(\alpha)$ attains a minimum at $\alpha = 0$ over the inverval $[0,1]$ so $\frac{dg(0)}{d\alpha} \geq 0$. But $\frac{dg(0)}{d\alpha} = \nabla f(x^*)'(x - x^*)$ (using the chain rule) so (7) is proved.

Conversely assume that (7) holds and x^* is not an optimal solution

of (6). We will arrive at a contradiction. Indeed let $\tilde{x}\epsilon X$ be such that $f(\tilde{x}) < f(x^*)$ and consider the function $g(\alpha) = f[x^* + \alpha(\tilde{x}-x^*)]$. Then $\frac{dg(0)}{d\alpha} \geq 0$ [by (7)] while $f(x^*) = g(0) > g(1) = f(\tilde{x})$. A little thought or an elementary argument shows that these conditions contradict the convexity of $g(\alpha)$ over $[0,1]$ and hence also the convexity of f. Q.E.D.

We now apply the lemma to problem (5). The lemma is applicable since both the objective function and the constraint set of (5) are convex. If h denotes the vector of the path flows h_p, D(h) denotes the objective function of problem (5) and $\frac{\partial D(h)}{\partial h_p}$ denotes the partial derivative of D with respect to h_p we see that

$$\frac{\partial D(h)}{\partial h_p} = \sum_{(i,k)\epsilon p} D'_{ik} \tag{8}$$

where the derivatives D'_{ik} are evaluated at the total flows corresponding to h. From (8) we see that $\partial D/\partial h_p$ is the length of the path p when length of each link (i,k) is taken to be the first derivative D'_{ik} evaluated at $\underline{h}$. According to the lemma $\{h_p^*\}$ is an optimal set of path flows if it satisfies the constraints of problem (5) and condition (7) is satisfied. By using (8), condition (7) can be written as

$$\sum_{w\epsilon P_w} \sum_{p\epsilon P_w} d_p^*(h_p - h_p^*) \geq 0 \tag{9}$$

for all h_p satisfying the constraints

$$\sum_{p\epsilon P_w} h_p = r_w, \quad h_p \geq 0, \qquad \forall\ p\epsilon P_w,\ w\epsilon W,$$

where d_p^* is the 1st derivative length of the path p given by

$$d_p^* = \sum_{(i,k)\in p} D'_{ik} = \frac{\partial D(h^*)}{\partial h_p}. \qquad (11)$$

Conditions (9) and (10) can be clearly decoupled with respect to OD pair and written for each $w\in W$ as

$$\sum_{p\in P_w} d_p^*(h_p - h_p^*) \geq 0, \quad \forall\, h_p \geq 0,\ p\in P_w \text{ with } \sum_{p\in P_w} h_p = r_w. \qquad (12)$$

It is easily seen (argue by contradiction) that this condition is equivalent to having for all $w\in W$

$$h_p^* > 0 \Longrightarrow d_p^* = \min_{p\in P_w} \{d_p^*\} \qquad (13)$$

Equivalently we have that a set of path flows is optimal if and only if path flow is positive only on paths with minimum 1st derivative length.

5. Shortest Path Routing and the Frank-Wolfe Method

We have seen that optimal routing results only if flow travels along minimum first derivative length (MFDL for short) for each OD pair. Equivalently a routing (i.e. a set of routing variables) is strictly suboptimal only if there is a positive amount of path flow that travels on a non MFDL path. This suggests that suboptimal routing can be improved by shifting flow to an MFDL path from other paths for each OD pair. Indeed this can be shown mathematically by observing that if $h = \{h_p\}$ is a set feasible path flows and $\Delta h = \{\Delta h_p\}$ is a corresponding direction for changing h then the function of the scalar α given by

$$G(\alpha) = D(h + \alpha\Delta h) \tag{1}$$

has first derivative

$$\left.\frac{dG(\alpha)}{d\alpha}\right|_{\alpha=0} = \sum_{w\in W} \sum_{p\in P_w} \frac{\partial D(h)}{\partial h_p} \Delta h_p \tag{2}$$

$$= \sum_{w\in W} \sum_{p\in P_w} d_p \Delta h_p$$

where d_p is the first derivative length of the path p (evaluated at the link flows corresponding to h). Therefore if Δh_p is positive for MFDL paths and negative for all other paths while maintaining the conservation of OD pair input flow equation

$$\begin{aligned} &\sum_{p\in P_w} \Delta h_p = 0 \quad , \quad \forall\ w\in W, \\ &h + \Delta h_p \geq 0 \quad , \quad \forall\ p\in P_w,\ w\in W \end{aligned} \tag{3}$$

we will have

$$\left.\frac{dG(\alpha)}{d\alpha}\right|_{\alpha=0} < 0 \tag{4}$$

which means that the objective function will be reduced by a small motion in the direction Δh.

The preceding discussion suggests the following iterative algorithm:

Given $h = \{h_p\}$ find a MFDL path for each OD pairs. Let $\bar{h} = \{\bar{h}_p\}$ be the set of path flows that would result if all input r_w for each OD pair $w \varepsilon W$ is routed along the corresponding MFDL path. Let α^* be the stepsize that minimizes $D[h + \alpha(\bar{h}-h)]$ over all $\alpha \varepsilon [0,1]$, i.e.

$$D[h + \alpha^*(\bar{h}-h)] = \min_{\alpha \varepsilon [0,1]} D[h + \alpha(\bar{h}-h)]. \tag{5}$$

The new set of path flows is obtained by

$$h \leftarrow h + \alpha^*(\bar{h}-h) \tag{6}$$

and the process is repeated.

This algorithm is a special case of the so called Frank-Wolfe method for solving general nonlinear programming problems with convex constraint sets (see [4],[5]). It has been called the flow deviation method (see [6]), and can be shown to reduce the value of the objective function to its minimum in the limit although its convergence rate near the optimum tends to be very slow. Proving convergence depends on selecting a proper value for the stepsize α. The determination of an optimal stepsize α^* satisfying (5) requires a one-dimensional mini-

mization over [0,1] which can be carried out through any one of several existing algorithms. However finding α^* constitutes an iterative process which makes the algorithm impossible to implement in a distributed manner. A simpler method is to choose the stepsize α^* in (6) by means of the formula

$$\alpha^* = \min\left[1, -\frac{\sum_{(i,k)} D'_{ik}(\bar{F}_{ik}-F_{ik})}{\sum_{(i,k)} D''_{ik}(\bar{F}_{ik}-F_{ik})^2}\right] \tag{7}$$

where $\{F_{ik}\}$ and $\{\bar{F}_{ik}\}$ are the sets of total link flows corresponding to $\{h_p\}$ and $\{\bar{h}_p\}$ respectively, and the first and second derivatives D'_{ik}, D''_{ik} are evaluated at F_{ik}. The formula (7) for α^* is obtained by making a second order Taylor series expansion $\tilde{G}(\alpha)$ of $G(\alpha) = D[\bar{h} + \alpha(\bar{h}-h)]$ around $\alpha = 0$

$$\tilde{G}(\alpha) = \sum_{(i,k)} \{D_{ik}(F_{ik}) + \alpha D'_{ik}(F_{ik})(\bar{F}_{ik}-F_{ik}) + \frac{\alpha^2}{2} D''_{ik}(F_{ik})(\bar{F}_{ik}-F_{ik})^2\}$$

and minimizing $\tilde{G}(\alpha)$ with respect to α over the interval [0,1].

It can be shown that the Frank-Wolfe algorithm (6) with the choice (7) for the stepsize converges to the optimal set of total link flows provided the starting set of total link flows is sufficiently close to the optimal. For the type of objective functions used in routing problems it appears that the stepsize choice (7) typically leads to convergence even when the starting total link flows are far from

optimal.

Aside from its simplicity the stepsize rule (7) has the advantage that it can be implemented in a distributed way by means of a scheme such as the one described below:

Each node i broadcasts the current value of F_{ik} for all of its outgoing links (i,k) to all other nodes. (This can be done by flooding or through a spanning tree). Each node calculates $D'_{ik}(F_{ik})$ and $D''_{ik}(F_{ik})$ for all links (i,k) and computes an MFDL path for each OD pair w for which it is the origin. It then sends the value r_w along the MFDL path. The head node of each link (i,k) adds up the inputs r_w for all the MFDL paths that go through it, computes the total flow $\overline{F}_{ik}$ and then broadcasts the values of $(\overline{F}_{ik}-F_{ik})$ to all other nodes. All nodes then can compute the stepsize a^* of (7) and compute the required change in path flows

$$h_p \leftarrow h_p + \alpha^* (\overline{h}_p - h_p)$$

and corresponding change in the path routing variables. The scheme requires two messages (F_{ik}, and $\overline{F}_{ik}$) per link to be broadcast to all nodes and one message per OD pair (r_w) to be sent to every node along the corresponding MFDL path. The communication complexity per iteration is $O(LN) + O(N^3)$ if a spanning tree is employed for broadcasting and $O(L^2) + O(N^3)$ if flooding is employed where N and L is the number of nodes and links respectively. We will describe in the next section other distributed optimal routing algorithms with better communication complexity per iteration and a typically better rate of convergence

than the Frank-Wolfe method.

There are several shortest path routing algorithms used in practice (see [7]) that resemble to some extent the Frank-Wolfe method although they fail to achieve optimality in any identifiable sense and in some cases they don't even come close to doing so. Their general form is as follows:

(SP) At discrete times an MFDL path is computed for each OD pair and all new traffic (datagrams or virtual circuits) generated in the intervening time period is routed along these MFDL paths.

The scheme above presupposes link lengths that are flow dependent and represent first derivatives of some other functions. Several shortest path routing algorithms used in practice employ link lengths that depend in a crude (and discontinuous) manner on the flow traversing the link. In some cases link lengths are taken to be constant (which corresponds to linear functions D_{ik}) and change only if the link fails in which case its length is set to (essentially $+\infty$). The ARPANET algorithm [8] uses as link length a time average of packet delay in traversing the link during the preceding time period.

The performance of algorithm (SP) strongly depends on the choice of the link function D_{ik} and its first derivative D'_{ik}, on the frequency of routing variable updates, and on the rate at which new traffic is generated in the network. If datagrams are used exclusively in the network, algorithm (SP) cannot possibly provide optimal or near optimal routing. Since there is no restriction for each datagram of a conversation to follow the same path as a previous datagram, algorithm

(SP) induces a very abrupt shift of flow when a currently used MFDL path is changed. As a result, at any given time, each OD pair communicates along a single path, and this is inconsistent with optimal routing where it is typically necessary to bifurcate flow at strategic points in order to avoid overloading some portions of the network relative to others. Furthermore shortest path routing in datagram networks can exhibit an oscillatory behavior whereby not only the MFDL paths change frequently but also an unfortunate tendency is exhibited by the algorithm to select shortest paths that are progressively worse with respect to any global congestion measure. An explanation and analysis of this phenomenon is given in [9].

Algorithm (SP) tends to work somewhat better in virtual circuit networks assuming that whenever an MFDL path update is made the virtual circuits in use are not switched over to the new path but continue using the same path as before. This in effect implies a gradual switch of traffic from the old MFDL path to the new one which may be viewed as an implementation of the Frank-Wolfe method. The amount of flow shift from the old MFDL paths to the new one corresponds to the stepsize used in the Frank-Wolfe method and basically depends on two factors:

a) The rate at which old conversations terminate and new conversations are generated and

b) The time interval between MFDL path updates.

It can be shown (as yet unpublished work) that this routing method tends to provide a sequence of routings that converges (rather slowly) to a neighborhood of the optimum and then oscillates within that neighborhood. The size of the neighborhood depends on the (effective) stepsize of the corresponding Frank-Wolfe method. As the stepsize decreases (slower rate of generation of new conversations, and faster MFDL path updates), the neighborhood becomes smaller.

In conclusion it may be said that shortest path routing bears some relation to optimal routing and the Frank-Wolfe method but it is often practiced in a way that can result in far from optimal performance. It makes more sense in virtual circuit networks but even for such networks its convergence to a neighborhood of an optimal solution tends to be slow.

6. Projection Methods for Optimal Routing

Methods in this category are also based on shortest paths and determine an MFDL path for every OD pair at each iteration. An increment of flow change is calculated for each path on the basis of the relative magnitudes of the path lengths and, sometimes, second derivatives of the objective function. If some path flow becomes negative on the basis of the corresponding flow increment it is simply set to zero, i.e. it is "projected" back onto the positive orthant. There are several methods of this type that are of interest in connection with the routing problem. They may all be viewed as constrained versions of common unconstrained optimization methods such as steepest descent and Newton's method extensive accounts of which may be found in any text on nonlinear programming, e.g. [4], [5], [13]. In what follows we describe briefly these methods in a general nonlinear optimization setting and subsequently specialize them to the routing problem.

Let $f: R^n \to R$ be a twice continuously differentiable convex function with gradient at any $x \in R^n$ denoted $\nabla f(x)$ and Hessian matrix $\nabla^2 f(x)$ assumed positive definite for all x. The method of steepest descent for finding an unconstrained minimum of f is given by the iteration

$$x_{k+1} = x_k - \alpha_k \nabla f(x_k) \quad , \quad k = 0,1,... \tag{1}$$

where α_k is a positive scalar stepsize determined according to some rule. Common choices for α_k are the minimizing stepsize determined

by

$$f[x_k - \alpha_k \nabla f(x_k)] = \min_{\alpha>0} f[x_k - \alpha \nabla f(x_k)] \tag{2}$$

and a constant positive stepsize $\bar{\alpha}$

$$\alpha_k \equiv \bar{\alpha} \quad , \quad \forall\, k. \tag{3}$$

There are a number of convergence results relating to method (1) with stepsize choices (2) or (3). For example if f has a unique unconstrained minimizing point it may be shown that the sequence $\{x_k\}$ generated by (1), (2) converges to this minimizing point for every starting x_0. Also given any starting vector x_0, the sequence generated by (1), (3) converges to the minimizing point provided $\bar{\alpha}$ is chosen sufficiently small. Unfortunately however the speed of convergence of $\{x_k\}$ can be quite slow. It can be shown [5], [13] that for the case of the line minimization rule (2) if f is a positive definite quadratic function

$$f(x) = \frac{1}{2} x^T Q x - b^T x,$$

where Q is a positive definite symmetric nxn matrix and b is a given vector, then there holds

$$\frac{f(x_{k+1})-f^*}{f(x_k)-f^*} \leq \left(\frac{M-m}{M+M}\right)^2 \quad , \; f^* = \min_x f(x) \tag{4}$$

where M and m are the largest and smallest eigenvalues of Q respectively. Furthermore there exist starting points x_0 such that (4) holds with equality for every k. So if the ratio M/m is large (this cor-

responds to the level sets of f being very elongated ellipses), the rate of convergence is slow. A similar result can be shown for the method (1) and (3) and these results can be shown to hold in a qualitatively similar form for general convex twice continuously differentiable functions f with everywhere positive definite Hessian matrix.

The rate of convergence of the steepest descent method can be improved by premultiplying the gradient by a suitable positive definite scaling matrix D_k thereby obtaining the iteration

$$x_{k+1} = x_k - \alpha_k D_k \nabla f(x_k) , \quad k = 0,1,... \tag{5}$$

From the point of view of rate of convergence the best method is obtained with the choice

$$D_k = [\nabla f(x_k)]^{-1}. \tag{6}$$

This is Newton's method which can be shown to possess a very fast (quadratic) speed of congergence near the minimizing point. Unfortunately this excellent speed of convergence is achieved at the expense of the potentially substantial overhead associated with the inversion operation in (6). It is often useful to consider other choices of D_k which approximate the "optimal" choice $[\nabla^2 f(x_k)]^{-1}$ but do not require as much computation overhead. A choice that often works well is to choose D_k to be a diagonal approximation to the inverse Hessian, i.e.

$$D_k = \begin{bmatrix} \left[\frac{\partial^2 f(x_k)}{(\partial x^1)^2}\right]^{-1} & & & 0 \\ & \left[\frac{\partial^2 f(x_k)}{(\partial x^2)^2}\right]^{-1} & & \\ & & \ddots & \\ 0 & & & \left[\frac{\partial^2 f(x_k)}{(\partial x^n)^2}\right]^{-1} \end{bmatrix} . \tag{7}$$

With this choice the method (5) can be written in the simple form

$$x^i_{k+1} = x^i_k - \alpha_k \left[\frac{\partial^2 f(x_k)}{(\partial x^i)^2}\right]^{-1} \frac{\partial f(x_k)}{\partial x^i}, \quad \begin{matrix} k = 0,1,... \\ i = 1,...n. \end{matrix} \tag{8}$$

Consider now the problem of minimizing the convex twice continuously differentiable function $f: R^n \to R$ subject to the nonnegativity constraints $x^i \geq 0$, $i = 1,...,n$, i.e. the problem

$$\begin{aligned} &\text{minimize} \quad f(x) \\ &\text{subject to } x \geq 0. \end{aligned} \tag{9}$$

A straightforward analog of the steepest descent method (1) is given by

$$x_{k+1} = [x_k - \alpha_k \nabla f(x_k)]^+, \quad k = 0,1,... \tag{10}$$

where for any vector $z \in R^n$, we denote by $[z]^+$ the projection of z onto the positive orthant

$$[z]^+ = \begin{bmatrix} \max\{0,z^1\} \\ \max\{0,z^2\} \\ \vdots \\ \max\{0,z^n\} \end{bmatrix}. \tag{11}$$

It can be shown [14] that the convergence results mentioned earlier in connection with the unconstrained steepest descent method (1) also hold true for the constrained analog (10). The same is true for the method

$$x_{k+1} = [x_k - \alpha_k D_k \nabla f(x_k)]^+ \tag{12}$$

where D_k is a diagonal positive definite scaling matrix such as (7) and for other rules of stepsize selection. While the assumption that D_k is diagonal is essential for the validity of iteration (12), there are modified versions of (12) in which D_k is chosen nondiagonal on the basis of the second derivatives of f and for which the fast convergence rate of Newton's method is realized. We will not consider these methods in these notes and we refer the reader to [16] and [15] for related description and analysis, as well as application to the routing problem. In what follows we concentrate on the application of the simple method (12) to the routing problem for D_k chosen to be a diagonal approximation to the inverse Hessian matrix.

Consider the routing problem in terms of path flows

$$\text{minimize} \sum_{(i,k)} D_{ik} \Big[\sum_{w \in W} \sum_{p \in P_w} \delta_p(i,k) h_p \Big] \triangleq D(h) \tag{13}$$

$$\text{subject to} \sum_{p \in P_w} h_p = r_w \ , \ h_p \geq 0, \quad \forall\, w \in W,\ p \in P_w$$

Assume that after k iterations we have a feasible set of path flows $\{h_p^k\}$, and let $\{F_{im}^k\}$ be the corresponding set of total link flows. For each OD pair w let $\bar{p}_w$ be an MFDL path [with respect to link lengths $D'_{im}(F_{im}^k)$]. We can convert problem (13) (for the purpose of the next iteration) to a problem involving only positivity constraints by expressing the flows of the MFDL paths $\bar{p}_w$ in terms of the other path flows while eliminating the equality constraints $\sum_{p \in P_w} h_p = r_w$ in the process. Thus we write for each $w \in W$

$$h_{\bar{p}_w} = r_w - \sum_{p \in P_w} h_p \tag{14}$$

and substitute $h_{\bar{p}_w}$ in the objective function D(h) thereby obtaining a problem of the form

$$\begin{aligned} &\text{minimize } \tilde{D}(\tilde{h}) \\ &\text{subject to } h_p \geq 0, \quad \forall\, w \in W,\ p \in P_w,\ p \neq \bar{p}_w \end{aligned} \tag{15}$$

where $\tilde{h}$ is the vector of all path flows which are <u>not</u> MFDL paths. The objective $\tilde{D}(\tilde{h})$ is obtained from D(h) once the MFDL path flows $h_{\bar{p}_w}$, $w \in W$ are substituted by their expressions (14) in terms of the other path flows. Clearly we have

$$\frac{\partial \tilde{D}(\tilde{h}^k)}{\partial h_p} = \frac{\partial D(h^k)}{\partial h_p} - \frac{\partial D(h^k)}{\partial h_{\bar{p}_w}}, \quad \forall\, p \in P_w,\ p \neq \bar{p}_w \tag{16}$$

for all $w \in W$. We have already seen in the previous section that $\frac{\partial D(h)}{\partial h_p}$ is the first derivative length of path p, i.e.

$$\frac{\partial D(h^k)}{\partial h_p} = \sum_{(i,m)\in p} D'_{im}(F^k_{im}). \tag{17}$$

Since $\bar{p}_w$ is an MFDL path we have from (16)

$$\frac{\partial \tilde{D}(\tilde{h}^k)}{\partial h_p} \geq 0, \quad \forall\, w\in W,\ p\in P_w,\ p \neq \bar{p}_w \tag{18}$$

Regarding second derivatives, a straightforward differentiation of the expression (16), (17) for the first derivative shows that

$$\frac{\partial^2 \tilde{D}(\tilde{h}^k)}{(\partial h_p)^2} = \sum_{(i,m)\in L_p} D''_{im}(F^k_{im}), \quad \forall\, w\in W,\ p\in P_w,\ p \neq \bar{p}_w \tag{19}$$

where, for each p, L_p is the set of links that belong to either the path p, or the corresponding MFDL path $\bar{p}_w$ but not both.

We now have available expressions for both first and second derivatives of the "reduced" objective function $\tilde{D}(\tilde{h})$ and thus we can apply the projection method (12) with the diagonal approximation of the inverse Hessian as scaling matrix. The iteration takes the form

$$h_p^{k+1} = \max\{0,\ h_p^k - \alpha_k L_p^{-1} (d_p - d_{\bar{p}_w})\} \quad \forall\, w\in W,\ p\in P_w,\ p \neq \bar{p}_w, \tag{20}$$

where d_p and $d_{\bar{p}_w}$ are the first derivative lengths of the paths p and $\bar{p}_w$ given by [cf. (17)]

$$d_p = \sum_{(i,m)\in p} D'_{im}(F^k_{im}),\quad d_{\bar{p}_w} = \sum_{(i,m)\in \bar{p}_w} D'_{im}(F^k_{im}) \tag{21}$$

and L_p is the "second derivative length"

$$L_p = \sum_{(i,m)\in L_p} D''_{im}(F^k_{im}) \tag{22}$$

given by (19).

The stepsize α_k is some positive scalar which may be chosen by a variety of methods. For example α_k could be chosen constant or by some form of line minimization. More about stepsize selection will be said later.

The following observations can be made regarding iteration (20):

a) Since for each OD pair $w \in W$ we have $d_p \geq d_{\bar{p}_w}$ for all $p \neq \bar{p}_w$ it follows that all path flows h_p, $p \neq \bar{p}_w$ which are positive will be reduced with the corresponding increment of flow being shifted to the MFDL path $\bar{p}_w$.

b) Those path flows h_p, $p \neq \bar{p}_w$ which are zero will stay at zero. Therefore the calculation indicated in (20) should only be carried out for paths that carry positive flow.

c) Only paths that carried positive flow at the starting flow pattern or were MFDL paths at some previous iteration can carry positive flow at the beginning of any single iteration. This is important in that it tends to keep the number of paths that carry positive flow small with a corresponding reduction in the amount of calculation and bookkeeping needed at each iteration.

Regarding the choice of the stepsize α_k there are several possibilities. It is possible to select α_k to be constant ($\alpha_k \equiv \alpha$, $\forall$ k), and with this choice it can be shown (the proof is essentially given in [17]) that given any starting set of path flows there exists $\bar{\alpha} > 0$ such that if for all k we have $0 < \alpha_k \leq \bar{\alpha}$ then a sequence generated by iteration (20)-(22) converges to the optimal value of the problem. A

crucial question has to do with the magnitude of the constant stepsize. It is known from nonlinear programming experience and analysis that a stepsize equal to unity usually works well with Newton's method as well as approximations to Newton's method that employ scaling based on second derivatives [5], [13]. Experience has verified that a choice of α_k in (20) near unity typically works quite well in iteration (20) regardless of the values of the input traffic pattern $\{r_w\}$. Even better performance with unity stepsize α_k is usually obtained if iteration (20) is carried out one OD pair (or one origin) at a time, i.e. first carry out (20) with $\alpha_k=1$ for a single OD pair (or origin) adjust the corresponding total link flows to account for the effected change in the path flows of this OD pair (or origin), and then carry out (20) with $\alpha_k=1$ for the next OD pair (or origin) until all path flows are taken up cyclically. The rationale for this is based on the fact that by dropping the off-diagonal terms of the Hessian matrix [cf. (5),(7)] we are in effect neglecting the interaction between the flows of different OD pairs. In other words iteration (20) is based to some extent on the premise that each OD pair will adjust its own path flows while the other OD pairs will keep theirs unchanged. By carrying out (20) one OD pair at a time we can reduce the potentially deterimental effect of the neglected off-diagonal terms of the Hessian and increase the likelihood that the unity stepsize is appropriate and effective. Under these circumstances iteration (20) works well with a unity stepsize for almost all networks and traffic input patterns likely to be encountered in practice.

Another possibility, which is better suited for a centralized

implementation is to select α_k by a simple form of line search in equation (20). Thus let $\{F_{ik}\}$ be the set of link flows corresponding to $\{h_p^k\}$ and let $\{\bar{F}_{ik}\}$ be the set of link flows corresponding to the set $\{\bar{h}_p\}$ given by [cf. (20) with $\alpha_k = 1$]

$$\bar{h}_p = h_p^k - L_p^{-1}(d_p - d_{\bar{p}_w}), \quad \forall\, w \in W,\ p \in P_w,\ p \neq \bar{p}_w$$

$$\bar{h}_{p_w} = r_w - \sum_{p \neq \bar{p}_w} \bar{h}_p$$

The stepsize α_k in (20) is chosen to minimize the 2nd order Taylor series expansion of the objective along the line segment connecting $\{F_{ik}\}$ and $\{\bar{F}_{ik}\}$, i.e. [cf. (7)]

$$\alpha_k = - \frac{\sum_{(i,k)} D'_{ik}(\bar{F}_{ik} - F_{ik})}{\sum_{(i,k)} D''_{ik}(\bar{F}_{ik} - F_{ik})^2} .$$

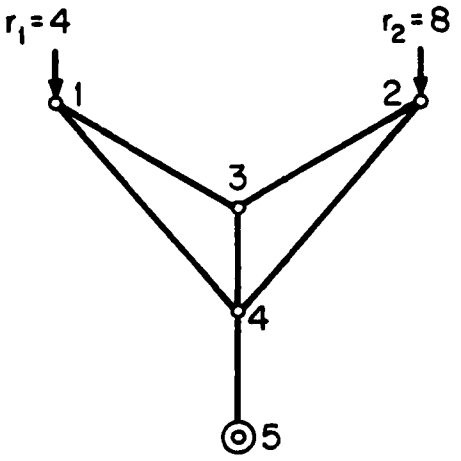
The algorithm (20) described above typically yields rapid convergence to a neighborhood of an optimal solution. Once it comes near a solution (how "near" is "near" depends on the problem) it tends to slow down. Its progress is often satisfactory near a solution and in any case far better than that of the Frank-Wolfe method.

In order for one to obtain fast convergence near a solution (and therefore also an accurate approximation to an optimal solution in a reasonable amount of time) it is necessary to take fully into account the off-diagonal terms of the Hessian matrix and introduce some form of line search for finding a proper stepsize. Surprisingly it is possible to implement sophisticated

methods of this type (see [15]) although we will not go into this further. We only mention that these more sophisticated methods are based on a more accurate approximation of a constrained version of Newton's method (using the conjugate gradient method) and attain the very fast rate of convergence of Newton's method near an optimal solution. However when far from a solution their speed of convergence is usually only slightly superior to that of iteration (20). So if one is only interested in getting fast near an optimal solution but the subsequent rate of progress is of little importance (as is typically the case in practical routing problems) the simple iteration (20) is usually fully satisfactory.

We now illustrate the algorithm (20)-(22) by means of an example:

Example: Consider the network shown in the figure below:



There are only two OD pairs (1,5) and (2,5) with corresponding inputs $r_1 = 4$, $r_2 = 8$ as shown in the figure. We consider the following two paths for each OD pair:

Paths of OD Pair (1,5):

$p_1(1) = \{(1,4),(4,5)\}$

$p_2(1) = \{(1,3), (3,4),(4,5)\}$

Paths of OD Pair (2,5):

$p_1(2) = \{(2,4), (4,5)\}$

$p_2(2) = \{(2,3), (3,4), (4,5)\}$

Consider the instance of the routing problem (13) where the link objective functions are all identical and given by

$$D_{ik}(F_{ik}) = \frac{1}{2}(F_{ik})^2 , \qquad \forall (i,k).$$

Consider an initial path flow pattern whereby each OD pair input is divided equally between the two available paths. This results in a flow distribution given in the following tables:

OD Pair	Path	Path Flow
(1,5)	$p_1(1)$	2
	$p_2(1)$	2
(2,5)	$p_1(2)$	4
	$p_2(2)$	4

Table 1

Link	Total Link Flow
(1,3)	2
(1,4)	2
(2,3)	4
(2,4)	4
(3,4)	6
(4,5)	12
others	0

Table 2

The first derivative length of each link is given by

$$D'_{ik}(F_{ik}) = F_{ik}$$

so the total link flows given in Table 2 are also the link lengths for the current iteration. The corresponding first derivative lengths of paths are given in the following table:

OD Pair	Path	1st Der. Length
(1,5)	$p_1(1)$	14
	$p_2(1)$	20
(2,5)	$p_1(2)$	16
	$p_2(2)$	22

Table 3

Therefore the shortest paths for the current iteration are $p_1(1)$ and $p_1(2)$ for OD pairs (1,5) and (2,5) respectively.

We now show the form of iteration (20)-(22) for each of the OD pairs:

OD Pair (1,5): Here for the nonshortest path $p = p_2(1)$ and the shortest path $\bar{p}_w = p_1(1)$ we have $d_p = 20$, $d_{\bar{p}_w} = 14$. We also have $L_p = 3$ [each link has second derivative length $D''_{ik} = 1$ and there are three links that belong to either $p_1(1)$ or $p_2(1)$ but not to both cf. (19),(22)]. Therefore iteration (20) takes the form

$$h_p \leftarrow \max\{0,\ 2 - \alpha_k \tfrac{1}{3}(20-14)\}$$

$$= \max\{0,\ 2 - 2\alpha_k\}$$

and

so the total link flows given in Table 2 are also the link lengths for the current iteration. The corresponding first derivative lengths of paths are given in the following table:

OD Pair	Path	1st Der. Length
(1,5)	$p_1(1)$	14
	$p_2(1)$	20
(2,5)	$p_1(2)$	16
	$p_2(2)$	22

Table 3

Therefore the shortest paths for the current iteration are $p_1(1)$ and $p_1(2)$ for OD pairs (1,5) and (2,5) respectively.

We now show the form of iteration (20)-(22) for each of the OD pairs:

OD Pair (1,5): Here for the nonshortest path $p = p_2(1)$ and the shortest path $\bar{p}_w = p_1(1)$ we have $d_p = 20$, $d_{\bar{p}_w} = 14$. We also have $L_p = 3$ [each link has second derivative length $D''_{ik} = 1$ and there are three links that belong to either $p_1(1)$ or $p_2(1)$ but not to both cf. (19),(22)]. Therefore iteration (20) takes the form

$$h_p \leftarrow \max\{0,\ 2 - \alpha_k \tfrac{1}{3}(20-14)\}$$

$$= \max\{0,\ 2 - 2\alpha_k\}$$

and

$$h_{\bar{p}_w} \leftarrow r_1 - h_p$$

$$= 4 - \max\{0, 2 - 2\alpha_k\}$$

OD Pair (2,5): Here for the nonshortest path $p = p_2(2)$ and the shortest path $\bar{p}_w = p_1(2)$ we have $d_p = 22$ and $d_{\bar{p}_w} = 16$. We also have $L_p = 3$ and iteration (20) takes the form

$$h_p \leftarrow \max\{0, 4 - \alpha_k \tfrac{1}{3}(22-16)\}$$

$$= \max\{0, 4 - 2\alpha_k\}$$

and

$$h_{\bar{p}_w} \leftarrow r_2 - h_p$$

$$= 8 - \max\{0, 4 - 2\alpha_k\}$$

More generally let $h_1(1)$, $h_2(1)$, $h_1(2)$, $h_2(2)$ denote the flows along the paths $p_1(1)$, $p_2(1)$, $p_1(2)$, $p_2(2)$, respectively at the beginning of the iteration. The corresponding path lengths are as follows:

$$d_{p_1(1)} = h_1(1) + r_1 + r_2$$

$$d_{p_2(1)} = 2h_2(1) + h_2(2) + r_1 + r_2$$

$$d_{p_1(2)} = h_1(2) + r_1 + r_2$$

$$d_{p_2(2)} = 2h_2(2) + h_2(1) + r_1 + r_2$$

The second derivative length L_p of (22) equals 3. The algorithm (20)-(22) takes the following form:

OD Pair (1,5):

If $d_{p_1(1)} > d_{p_2(1)}$

$$h_1(1) \leftarrow \max\{0,\ h_1(1) - \frac{\alpha_k}{3}\,[d_{p_1(1)} - d_{p_2(1)}]\}$$

$$h_2(1) \leftarrow r_1 - \max\{0,\ h_1(1) - \frac{\alpha_k}{3}\,[d_{p_1(1)} - d_{p_2(1)}]\}$$

If $d_{p_1(1)} \leq d_{p_2(1)}$

$$h_2(1) \leftarrow \max\{0,\ h_2(1) - \frac{\alpha_k}{3}\,[d_{p_2(1)} - d_{p_1(1)}]\}$$

$$h_1(1) \leftarrow r_1 - \max\{0,\ h_2(1) - \frac{\alpha_k}{3}\,[d_{p_2(1)} - d_{p_1(1)}]\}$$

OD Pair (2,5):

If $d_{p_1(2)} > d_{p_2(2)}$

$$h_1(2) \leftarrow \max\{0,\ h_1(2) - \frac{\alpha_k}{3}\,[d_{p_1(2)} - d_{p_2(2)}]\}$$

$$h_2(2) \leftarrow r_2 - \max\{0,\ h_1(2) - \frac{\alpha_k}{3}\,[d_{p_1(2)} - d_{p_2(2)}]\}$$

If $d_{p_1(2)} \leq d_{p_2(2)}$

$$h_2(2) \leftarrow \max\{0,\ h_2(2) - \frac{\alpha_k}{3}\,[d_{p_2(2)} - d_{p_1(2)}]\}$$

$$h_1(2) \leftarrow r_2 - \max\{0,\ h_2(2) - \frac{\alpha_k}{3}\,[d_{p_2(2)} - d_{p_1(2)}]\}$$

Notice that the presence of the link (4,5) does not affect at all the form of the iteration and indeed that should be so since

the total flow of link (4,5) is always equal to r_1+r_2 independently of the routing.

The following table gives sequences of successive objective function values obtained by the algorithm for different stepsize values, and the "all OD pair at once" and "one OD pair at a time" modes of implementation. The network topology is the same as above except that the inconsequential link (4,5) is deleted. The starting point for all runs is

$$h_1^0(1) = 0,\ h_2^0(1) = 4,\ h_1^0(2) = 0,\ h_2^0(2) = 8,$$

i.e. all flow is initially routed through the middle link (3,4) which is the worst possible starting flow pattern. The stepsize is chosen to be constant at one of three possible values ($\alpha_k \equiv 0.5$, $\alpha_k \equiv 1$, $\alpha_k \equiv 1.8$). It can be seen that for a unity stepsize the convergence to a neighborhood of a solution is very fast in both the one-at-a-time and the all-at-once modes. As the stepsize is increased the danger of divergence increases with divergence occuring typically first for the all-at-once mode. This can be seen from the table where for $\alpha_k \equiv 1.8$ the algorithm converges in the one-at-a-time mode but diverges in the all-at-once mode.

We finally mention two possible distributed implementations of iteration (20). One possibility is for all nodes i to broadcast to all other nodes the current total flows F_{im}^k of their outgoing links (i,m). Each node then computes the MFDL paths of OD pairs for which it is the origin and executes iteration (20) for some fixed stepsize.

Iteration k	All-at-Once			One-at-a-Time		
	$\alpha_k \equiv 0.5$	$\alpha_k \equiv 1.0$	$\alpha_k \equiv 1.8$	$\alpha_k \equiv 0.5$	$\alpha_k \equiv 1.0$	$\alpha_k \equiv 1.8$
0	112.0	112.0	112.0	112.0	112.0	112.00
1	38.88	32.00	40.00	42.44	29.33	40.00
2	30.39	29.33	46.72	31.54	29.00	37.15
3	29.28	29.03	40.00	29.63		29.79
4	29.09	29.00	46.72	29.29		29.56
5	29.03		40.00	29.08		29.33
6	29.01		46.72	29.03		29.18
7	29.00		40.00	29.01		29.12
8			46.72	29.00		29.09
9			40.00			29.06
10			46.72			29.03

Table 4

This corresponds to the "all OD pairs at once" mode of implementation and requires O(LN) or $O(L^2)$ link flow transmissions depending on whether a spanning tree or flooding is used for broadcasting.

The other possibility is for all nodes i to broadcast to a special node (say node 1) the current total flows F_{im}^k. This node computes the MFDL paths of OD pairs for which it is the origin and executes iteration (20) for a unity stepsize. It then computes the adjusted values of the total link flows taking into account the results of its own iteration and passes these values to a neighboring node who does the same thing until all nodes are taken up cyclically. This corresponds to the "one at a time" mode of implementation. It requires the same order of communication complexity as the "all at once" mode described earlier. For both implementation modes the communication complexity is more favorable than the one of the distributed implementation of the Frank-Wolfe method described in the previous section.

7. Combined Optimal Routing and Flow Control

While routing is concerned with the choice of good routes for messages (or other data units such as packets, virtual circuits, etc.) that have been accepted into the network, flow control deals with the question of whether particular messages (or other data units) should be allowed to enter the network. It is possible to consider several types of flow control in a data network depending on the points between which it is exercised (see [18] for a survey). Thus link-by-link (or hop level) flow control refers to procedures that limit the amount of flow from the headnode to the tailnode of a link. End-to-end flow control refers to procedures that limit the amount of flow that is input from external sources at an origin node of the communication subnetwork and is destined to another node, i.e. the input flows r_w introduced in Section 4 [cf. the routing problem (5)].

This section deals with the possibility of combining routing with end-to-end flow control by adjusting optimally both the routing variables as well as the inputs r_w. If the input r_w is measured in terms of virtual circuits, then its optimal value can be viewed as a target value that the origin node strives to achieve by blocking or allowing new calls generated from external sources. Similarly in integrated voice and data networks r_w can be related to rate of encoding of digitized voice and can be directly adjusted at the origin nodes (see [12]). When flow control is effected in terms of end-to-end windows (see [18]) there is some difficulty in determining window sizes that achieve the desired optimal inputs r_w. We refer the reader to [10], [11] for related discussion. In what follows we concentrate

on formulating a problem of adjusting routing variables together with the inputs r_w so as to minimize some "reasonable" objective function. We subsequently show that this problem is mathematically equivalent to the optimal routing problem examined in sections 4-6 (r_w: fixed) and therefore the optimality conditions and algorithms given there are applicable.

If we try to minimize the objective function $\sum_{(i,k)\in L} D_{ik}(F_{ik})$ used for the routing problem [cf. (5)] with respect to both the path flows $\{h_p\}$ and the inputs $\{r_w\}$, we unhappily find that the optimal solution is to set $h_p = 0$ and $r_w = 0$ for all p and w. This indicates that the objective function should be modified to include a penalty for the inputs r_w becoming too small and leads to the problem

$$\text{minimize} \quad \sum_{(i,k)\in L} D_{ik}\Big[\sum_{w\in W} \sum_{p\in P_w} \delta_p(i,k) h_p\Big] + \sum_{w\in W} e_w(r_w) \tag{23}$$

$$\text{subject to} \quad \sum_{p\in P_w} h_p = r_w, \quad \forall\, w\in W$$

$$h_p \geq 0, \quad \forall\, p\in P_w,\ w\in W$$

$$0 \leq r_w \leq \bar{r}_w, \quad \forall\, w\in W.$$

Here the minimization is to be carried out jointly with respect to $\{h_p\}$ and $\{r_w\}$. The given values $\bar{r}_w$ represent the amount of input desired by OD pair w, i.e. the maximum amount of input for w that would result if no flow control was exercised. The functions e_w are of the form shown below

and provide a penalty for throttling the inputs r_w. They are assumed to be convex, and monotonically decreasing on the half line $(0,\infty)$. We assume that their first and second derivatives e'_w and e''_w exist on $(0,\infty)$ and are strictly negative and positive respectively. An interesting class of functions e_w is specified by the following formula for their first derivative

$$e'_w(r_w) = -\left(\frac{a_w}{r_w}\right)^{b_w}, \quad a_w, b_w : \text{given positive constants}$$

As will be explained later in this section (see also [10]), the parameters a_w and b_w influence the magnitude of input r_w and the priority (relative magnitude of input allowed under heavy load conditions) of user class w respectively.

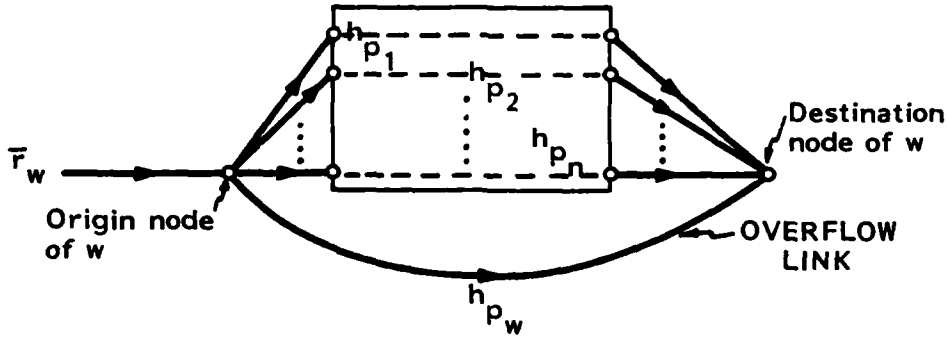
Similarly as for the routing problem h_p denotes the flow on path p, however it is important to note that some additional flexibility is provided by adopting a broader view of w and considering it as a class of users sharing the same set of paths P_w. This allows the possibility of providing different priorities (i.e. different functions e_w) to different classes of users even if they share the same paths. Furthermore it is possible to consider a problem where P_w consists

of a single path for each w in which case the routing component of the problem is essentially eliminated ($h_p = r_w$). A problem of strictly flow control results namely that of deciding upon the optimal fraction of the desired input flow of each user class that should be allowed into the network.

We now show that the combined routing and flow control problem (23) is mathematically equivalent to a routing problem of the type considered in Section 4 [cf. (5)]. Indeed let us introduce a new variable h_{p_w} for each $w \in W$ via the equation

$$r_w = \bar{r}_w - h_{p_w} \tag{25}$$

We may view h_{p_w} as the amount of <u>overflow</u> (portion of $\bar{r}_w$ blocked out of the network) and consider it as a flow on an <u>overflow link</u> p_w connecting directly the origin and destination nodes of w as shown below



If we define a new function $\bar{e}_w$ by

$$\bar{e}_w(h_w) = e_w(\bar{r}_w - h_{p_w}) \tag{26}$$

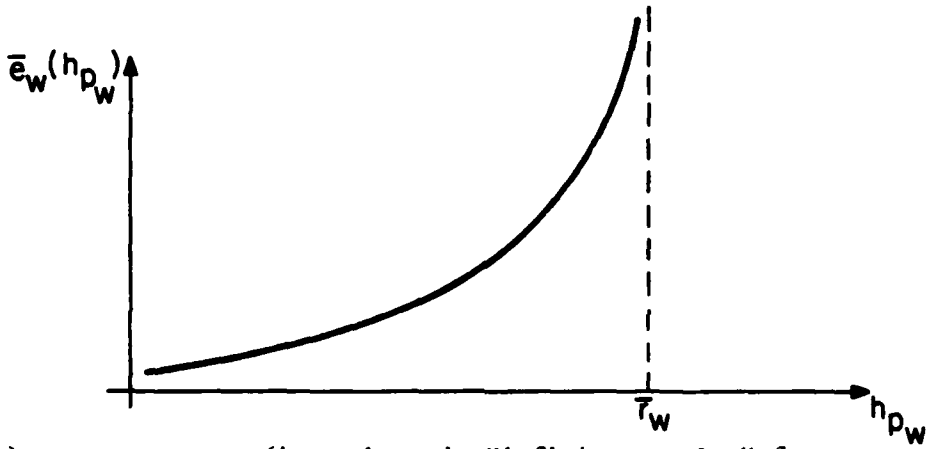
problem (23) becomes in view of (25)

$$\text{minimize} \quad \sum_{(i,k)\in L} D_{ik}\Big[\sum_{w\in W}\sum_{p\in P_w} h_p \delta_p(i,k)\Big] + \sum_{w\in W} \bar{e}_w(h_{p_w}) \qquad (27)$$

$$\text{subject to} \quad \sum_{p\in P_w} h_p + h_{p_w} = \bar{r}_w \,, \quad \forall\, w\in W$$

$$h_p \geq 0, \quad \forall\, p\in P_w,\ w\in W.$$

The form of the function $\bar{e}_w$ of (25) is shown below



If $e_w(r_w) \to \infty$ as $r_w \to 0$ (i.e. there is "infinite penalty" for completely shutting off the class of users w), then we have $\bar{e}_w(h_{p_w}) \to \infty$ as the overflow h_{p_w} approaches its maximum value--the maximum input $\bar{r}_w$. So we may view $\bar{e}_w$ as a "delay" function for the overflow link and consider $\bar{r}_w$ as the "capacity" of the link.

It is now clear that problem (27) is of the type considered in Sections 4-6 and that the algorithms and optimality conditions given there apply. In particular application of the optimality conditions of Section 4 yields the following result [cf. (13)]:

A feasible set of path flows $\{h_p^*\}$ and inputs $\{r_w^*\}$ is optimal

for problem (23) if and only if the following conditions hold for each $w \in W$:

$$h_p^* > 0,\ p \in P_w \Longrightarrow d_p^* = \min_{p \in P_w} \{d_p^*\},\quad d_p^* \leq -e_w'(r_w^*) \tag{28a}$$

$$r_w^* < \bar{r}_w \Longrightarrow -e_w'(r_w^*) = \min_{p \in P_w} \{d_p^*\} \tag{28b}$$

where d_p^* is the first derivative length of path p [$d_p^* = \sum_{(i,k) \in p} D_{ik}'(F_{ik}^*)$, and F_{ik}^* is the total flow of link (i,k) corresponding to $\{h_p^*\}$, cf. (11)].

The meaning of the parameters a_w and b_w in the objective function specified by the formula [cf. (24)]

$$e_w'(r_w) = -\left(\frac{a_w}{r_w}\right)^{b_w} \tag{29}$$

can now be made clear in the light of the optimality conditions (28). Consider two distinct classes of users w_1 and w_2 sharing the same paths ($P_{w_1} = P_{w_2}$). Then the conditions (28) imply that at an optimal solution in which both classes of users are throttled ($r_{w_1}^* < \bar{r}_{w_1}$, $r_{w_2}^* < \bar{r}_{w_2}$) we have

$$-e_{w_1}'(r_{w_1}^*) = -e_{w_2}'(r_{w_2}^*) = \min_{p \in P_{w_1}} \{d_p^*\} = \min_{p \in P_{w_2}} \{d_p^*\}$$

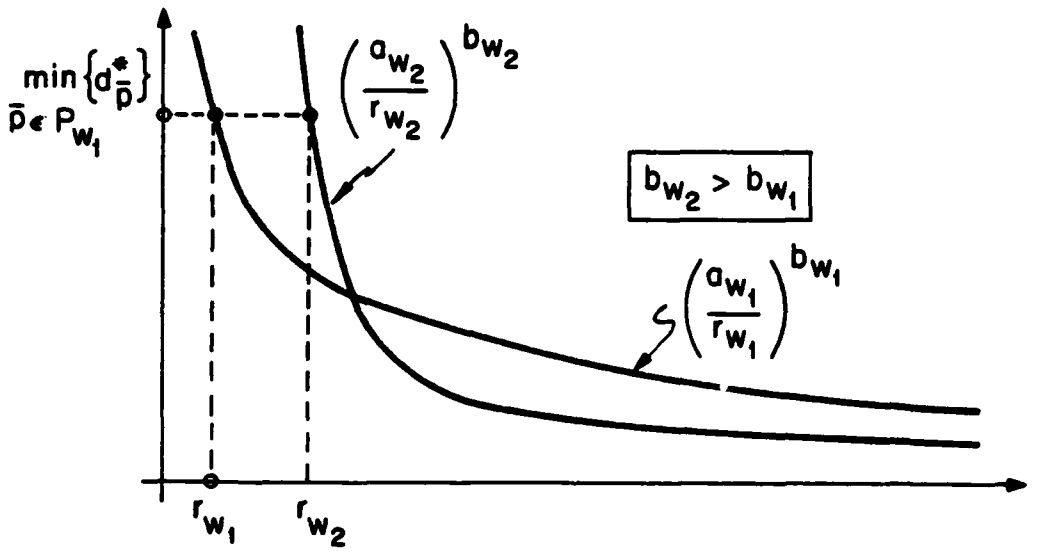
If e_{w_1}' and e_{w_2}' are specified by parameters a_{w_1}, b_{w_1} and a_{w_2}, b_{w_2} as in (29) we see that:

a) If $b_{w_1} = b_{w_2}$ then

$$\frac{r^*_{w_1}}{r^*_{w_2}} = \frac{a_{w_1}}{a_{w_2}}$$

and it follows that the parameter a_w influences the optimal relative input rate of the user class w.

b) If $a_{w_1} = a_{w_2}$ and $b_{w_1} < b_{w_2}$ (see the figure below)



then the condition (30) specifies that under heavy load conditions ($r^*_{w_1}$, $r^*_{w_2}$: small) the user class w_2 (the one with higher parameter b_w) will be allowed a larger input. It follows that the parameter b_w influences the relative priority of the user class w under heavy load conditions.

Optimal solutions of problem (23) possess several interesting properties. We refer to [10]-[12] for a more complete discussion. The reader may wish to verify as an exercise that the set of optimal

$\{r^*_w\}$ is unique (although the set of optimal $\{h^*_p\}$ need not be unique). Furthermore if $r^*_w < \bar{r}_w$, i.e. a positive amount of input for w is throttled, then the optimal input r^*_w will not change if $\bar{r}_w$ is increased. This means that the optimal input r^*_w is insensitive to increased demand from the user class w beyond a certain threshold.

References

[1] R. G. Gallager, "A Minimum Delay Routing Algorithm Using Distributed Computation", IEEE Trans. on Communications, Vol. COM-25, 1977, pp. 73-85.

[2] D. P. Bertsekas, E. M. Gafni, and R. G. Gallager, "Second Derivative Algorithms for Minimum Delay Distributed Routing in Networks", LIDS Report P-1082, M.I.T., March 1981.

[3] K. S. Vastola, "A Numerical Study of Two Measures of Delay for Network Routing", M.S. Thesis, Dept. of Electrical Engineering, Univ. of Illinois, Urbana, Sept. 1979.

[4] W. Zangwill, Nonlinear Programming: A Unified Approach, Prentice Hall, 1969.

[5] D. P. Bertsekas, "Notes on Nonlinear Programming and Discrete-Time Optimal Control", LIDS Report R-919, M.I.T., Aug. 1981.

[6] L. Fratta, M. Gerla, and L. Kleinrock, "The Flow Derivation Method: An Approach to Store-and-Forward Communication Network Design", Networks, Vol. 3, 1973, pp. 97-133.

[7] M. Schwartz and T. E. Stern, "Routing Techniques Used in Computer Communication Networks", IEEE Trans. on Communications, Vol. COM-28, 1980, pp. 539-552.

[8] J. M. McQuillan, I. Richer, and E. C. Rosen, "The New Routing Algorithm for the ARPANET", IEEE Trans. on Communications, Vol. COM-28, 1980, pp. 711-719.

[9] D. P. Bertsekas, "Dynamic Behavior of Shortest Path Routing Algorithm for Communication Networks", LIDS Report, 1980, to appear in IEEE Trans. on Aut. Control.

[10] R. G. Gallager and S. J. Golestaani, "Flow Control and Routing Algorithms for Data Networks", Proc. of the Fifth International Conference on Computer Communication (ICCC-80), Atlanta, Ga., Oct. 1980, pp. 779-784.

[11] S. J. Golestaani, "A Unified Theory of Flow Control and Routing in Data Communication Networks", LIDS Report TH-983, M.I.T., Jan. 1980.

[12] O. C. Ibe, "Flow Control and Routing in an Integrated Voice and Data Communication Network", LIDS Report TH-1115, M.I.T., August, 1981.

[13] D. G. Luenberger, Introduction to Linear and Nonlinear Programming, Addison-Wesley, Reading, Mass., 1973.

[14] D. P. Bertsekas, "On the Goldstein-Levitin-Poljak Gradient Projection Method", IEEE Trans. on Aut. Control, Vol. AC-21, 1976, pp. 174-184.

[15] D. P. Bertsekas and E. M. Gafni, "Projected Newton Methods and Optimization of Multicommodity Flows", LIDS Report P-1140, M.I.T., Aug. 1981.

[16] D. P. Bertsekas, "Projected Newton Methods for Optimization Problems with Simple Constraints", LIDS Report R-1025, M.I.T., Aug. 1980, to appear in SIAM J. on Control and Optimization.

[17] D. P. Bertsekas and E. M. Gafni, "Projection Methods for Variational Inequalities with Application to the Traffic Assignment Problem", LIDS Report P-1043, September 1980, to appear in Math. Programming Studies.

[18] M. Gerla and L. Kleinrock, "Flow Control: A Comparative Survey", IEEE Trans. on Comm., Vol. COMM-28, 1980, pp. 553-574.